EXP06-PL. Do not use an array in an implicit scalar context

Perl has two contexts in which expressions can be evaluated: scalar and list. These contexts determine what the expression generates. It is recommended that the context be made explicit when an expression is evaluated in an unexpected context. Implicit context switching makes programs difficult to read and more error prone.

Noncompliant Code Example

This noncompliant code example tries to print out the elements of an array.

```perl
sub print_array {
    my @array = shift;
    print "( ",
    foreach $item (@{ $array }) {
        print "$item , ";
    }
    print ")\n";
    my @array; # initialize
    my $array_ref = @array;
    print_array( $array_ref);
}
```

The developer mistakenly left out the \ indicator when initializing $array_ref. Consequently, instead of a reference to the array, it contains the number of elements in the array. When passed to the `print_array()` subroutine, this program prints an empty array.

Compliant Solution

This compliant solution initializes $array_ref correctly.

```
my $array_ref = \@array;
print_array( $array_ref);
```

Noncompliant Code Example

This noncompliant code example prints the number of elements in an array.

```perl
my @array; # initialize
my $cardinality = @array;
print "The array has $cardinality elements\n";
```

Although this program works correctly, the number of elements of an array can be obtained in less ambiguous ways.

Compliant Solution (`scalar()`)

This compliant solution uses the `scalar()` built-in subroutine to obtain the number of elements of an array.

```
my $cardinality = scalar( @array);
print "The array has $cardinality elements\n";
```

This compliant solution evaluates `@array` in scalar context just as in the noncompliant code example. However, the `scalar()` makes this evaluation explicit, removing any doubt as to the programmer's intentions.

Compliant Solution ($#)

This compliant solution uses the `#$` operator to obtain the number of elements of an array.

```
my $cardinality = @$array + 1;
print "The array has $cardinality elements\n";
```
Risk Assessment

Evaluating an array or hash in improper contexts can lead to unexpected and surprising program behavior.

<table>
<thead>
<tr>
<th>Recommendation</th>
<th>Severity</th>
<th>Likelihood</th>
<th>Remediation Cost</th>
<th>Priority</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXP06-PL</td>
<td>low</td>
<td>unlikely</td>
<td>medium</td>
<td>P2</td>
<td>L3</td>
</tr>
</tbody>
</table>

Automated Detection

<table>
<thead>
<tr>
<th>Tool</th>
<th>Diagnostic</th>
</tr>
</thead>
<tbody>
<tr>
<td>B::Lint</td>
<td>context</td>
</tr>
</tbody>
</table>

Bibliography

[Beattie]

[CPAN] Elliot Shank, Perl-Critic-1.116, ProhibitBooleanGrep


[Wall 2011] perlfunc